IBM Developer
SKILLS NETWORK

Winning Space Race with Data Science

Tomas L Ashenafi
23 June 2023

Outline

- Executive Summary
- Introduction
- Methodology
- Results
- Conclusion
- Appendix

Executive Summary

- Two methods were used to collect data. The first method used “requests” API to retrieve JSON formatted data and converted it to a pandas data frame. The second method employed web scrapping of html formatted data via “bs4” library.
- Applied exploratory data analysis to find meaningful patterns in the data and determine what would be the label for training supervised models. Discovered interesting relationship between features.
- Applied SQL and visual aid to perform a deeper exploratory data analysis. Helped us understand some details about key events and features.
- Took visual aid one step further by building interactive maps, plots and charts. This brought words to life.
- Applied machine learning libraries to build, tune and evaluate classification models. Revealed Decision trees classifier as the best estimator based on a scoring method and confusion matrix.

Introduction

The project applies data science methodologies and tools acquired in the IBM Data Science professional certification to define and solve a real-world business problem for an emerging space company called Space-Y. The company would like to know if it stands any chance of winning government contract to carry payload against Space-X which advertises its falcon 9 launches for around $62 Million compared to most competitions which have a price tag of over $162 Million due to the rocket's re-usability.
This was primarily accopmplished through predicting if the first stage of the Falcon 9 rocket will land successfully on its next mission based on a publicly available data on the Space-X website.

Section 1

Methodology

Methodology

Executive Summary

- Data collection:
  - Two methods used. The first method used “requests” API to retrieve JSON formatted data and converted it to a pandas data frame. The second method employed web scrapping of html formatted data via “bs4” library .
- Data wrangling:
  - Applied exploratory data analysis to find meaningful patterns in the data and determine what would be the label for training supervised models.
- Exploratory data analysis (EDA) using visualization and SQL:
  - Applied SQL and visual aid to perform a deeper exploratory data analysis.
- Interactive visual analytics using Folium and Plotly - Dash:
  - Took visual aid one step further by building interactive maps, plots and charts.
- Predictive analysis using classification models:
  - Applied machine learning libraries to build, tune and evaluate classification models

Data Collection

- The data collection process applied two alternative methods. The first method used “requests” API to retrieve JSON formatted data and converted it to a pandas data frame.

- The second method employed scrapping of html formatted data from a Wikipedia page via the request API. An html parsing library known as beautiful soup together with Pandas library were used to format the data.

Data Collection – SpaceX API

- Import Libraries and Define Auxiliary Functions such as requests, pandas, numpy, datetime
- Request and parse the SpaceX launch data using the GET request
  - For each placeholder column in our dataframe define a request object
- Filter the dataframe to only include Falcon 9 Launches
- Perform light data wrangling by dealing with Missing Values

- https://github.com/Tomas61692/IBM-Data-Science-Capstone/blob/main/JSON%20Data%20collection%20API.ipynb

Data Collection - Scraping

- Imported Libraries and Define Auxiliary Functions such as requests, pandas, sys, bs4, re,…
- For each placeholder column in dataframe defined a request object
- Requested the Falcon9 Launch Wiki page from its URL
- Extracted all column/variable names from the HTML table header
- Created a data frame by parsing the launch HTML tables

- https://github.com/Tomas61692/IBM-Data-Science-Capstone/blob/main/HTML%20data%20webscraping.ipynb

Data Wrangling

- Import necessary libraries such as fetch, pandas, io, numpy
- Identify and calculate the percentage of the missing values in each attribute
- Calculate the number of launches on each site
- Calculate the number and occurrence of each orbit
- Calculate the number and occurence of mission outcome per orbit type
- Create a landing outcome label from Outcome column

- https://github.com/Tomas61692/IBM-Data-Science-Capstone/blob/main/data%20wrangling.ipynb

EDA with Data Visualization

- First, explored possible patterns using seaborn to produce category plots of Flight Number vs payload mass, FlightNumber vs LaunchSite, Payload Vs. Launch Site, FlightNumber vs Orbit type, Payload vs Orbit type.
- Also related a bar plot of orbit vs class (launch outcome)
- After creating a function to Extract years from the date, used a line plot for the date vs class.

- https://github.com/Tomas61692/IBM-Data-Science-Capstone/blob/main/EDA%20with%20data%20Vizn.ipynb

EDA with SQL

- Imported the relevant libraries and downloadd the datasets
- Displayed the names of the unique launch sites in the space mission
- Displayed 5 records where launch sites begin with the string 'CCA’
- Displayed the total payload mass carried by boosters launched by NASA (CRS)
- Displayed average payload mass carried by booster version F9 v1.1
- Listed the date when the first succesful landing outcome in ground pad was acheived.
- Listed the names of the boosters which have success in drone ship and have payload mass greater than 4000 but less than 6000
- Listed the total number of successful and failure mission outcomes
- Listed the names of the booster_versions which have carried the maximum payload mass. Use a subquery
- Listed the records which will display the month names, failure landing_outcomes in drone ship ,booster versions, launch_site for the months in year 2015.

- https://github.com/Tomas61692/IBM-Data-Science-Capstone/blob/main/EDA%20with%20SQL.ipynb

Build an Interactive Map with Folium

- Created a site map object with NASA JSC as its focus location.
- Used folium.Circle, folium.Circle, folium.Popup to add a highlighted circle area with a text label, blue marker and popup text on NASA's JSC coordinate.
- Applied similar methodology to mark each launch site on the site map object.
- Applied a marker cluster object to place launch IDs that took place in the same launch site.
- Added a MousePosition on the map to get coordinate for a mouse over a point on the map
- Calculated the distances between a launch site to its proximities. This is done to compare the ease of access of each launch site to a nearby infrastructure.
- Haversine function was used to accomplish distance calculation and folium.PolyLine was used to draw a line between the locations.

- https://github.com/Tomas61692/IBM-Data-Science-Capstone/blob/main/Folium_Interactive_Map.ipynb

Build a Dashboard with Plotly Dash

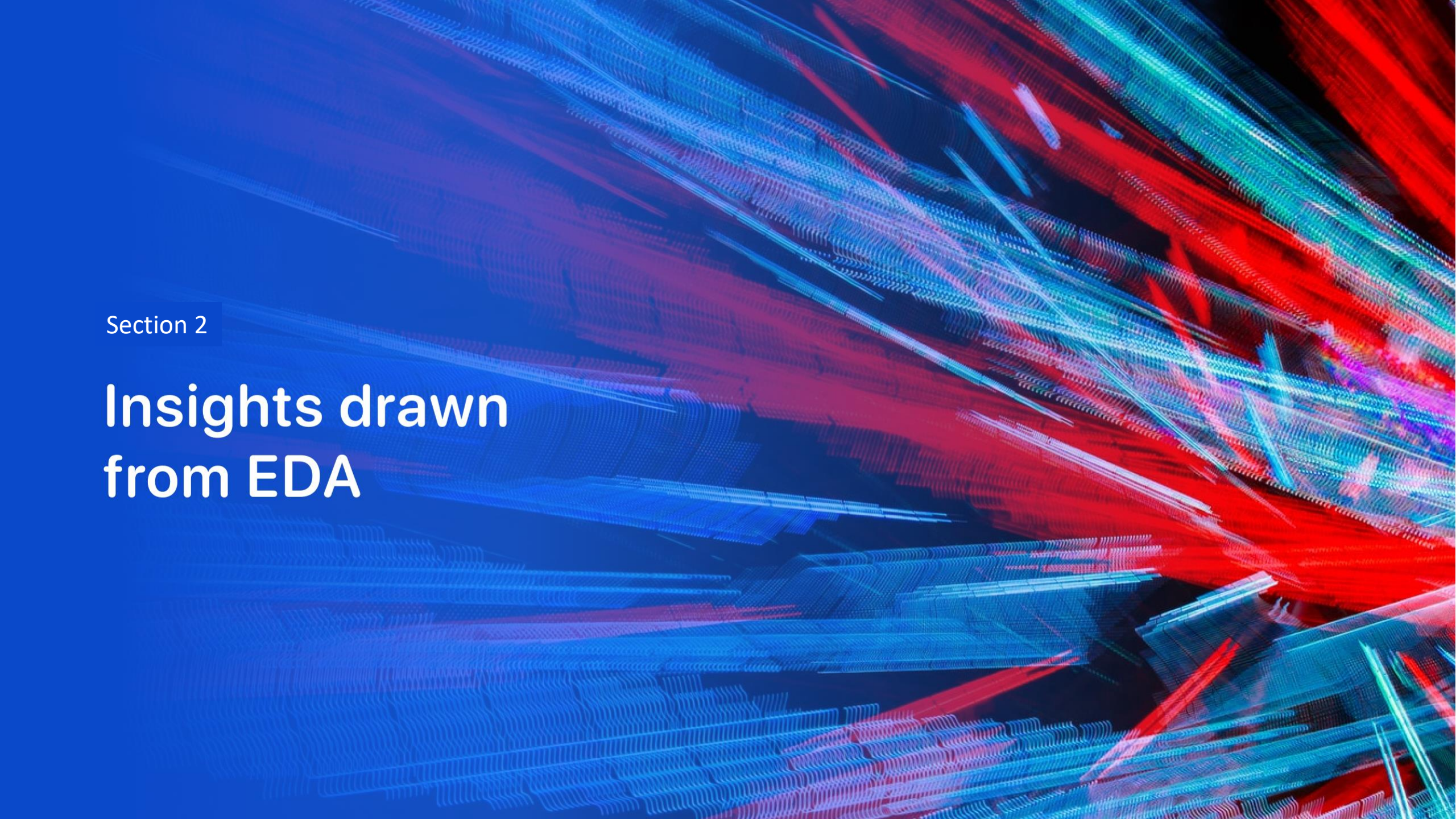
- Added a Launch Site Drop-down Input Component to see which one has the largest success count and to select one specific site and check its detailed success rate (class=0 vs. class=1).
- Added a callback function to render success-pie-chart based on selected site dropdown
- Added a Range Slider to Select Payload to find if variable payload is correlated to mission outcome and be able to easily select different payload range and see if we can identify some visual patterns
- Added a callback function to render the success-payload-scatter-chart scatter plot

- https://github.com/Tomas61692/IBM-Data-Science-Capstone/blob/main/Interactive_Web_dashboard.ipynb

Predictive Analysis (Classification)

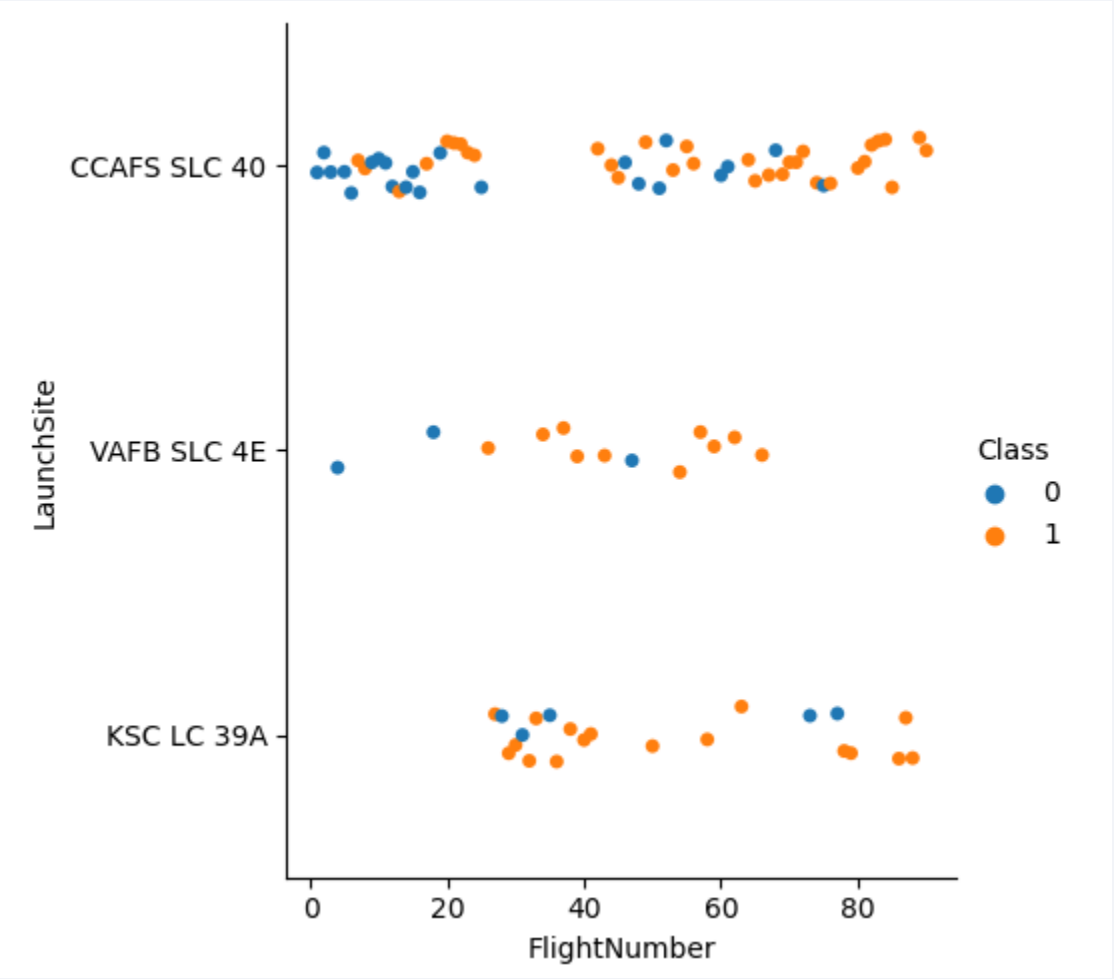
- Created a NumPy array from the column Class in data, by applying the method to_numpy() then assign it to the variable Y.
- Standardized the data in X then reassign it to the variable X
- Used the function train_test_split to split the data X and Y into training and test data.
- Created logistic regression, decision tree classifer, KNearestNeighbors, and support vector machines objects independently.
- in each case created a GridSearchCV object logreg_cv with cv = 10.
- In each case fited each object to find the best parameters from the associated dictionary parameters.
- In each case calculated the accuracy on the test data using the method score
- In each case plotted a confusion matrix.
- Eventually made a comparison between the model to determine the best estimator among them.

- https://github.com/Tomas61692/IBM-Data-Science-Capstone/blob/main/Machine_Learning.ipynb


Section 2
Insights drawn from EDA

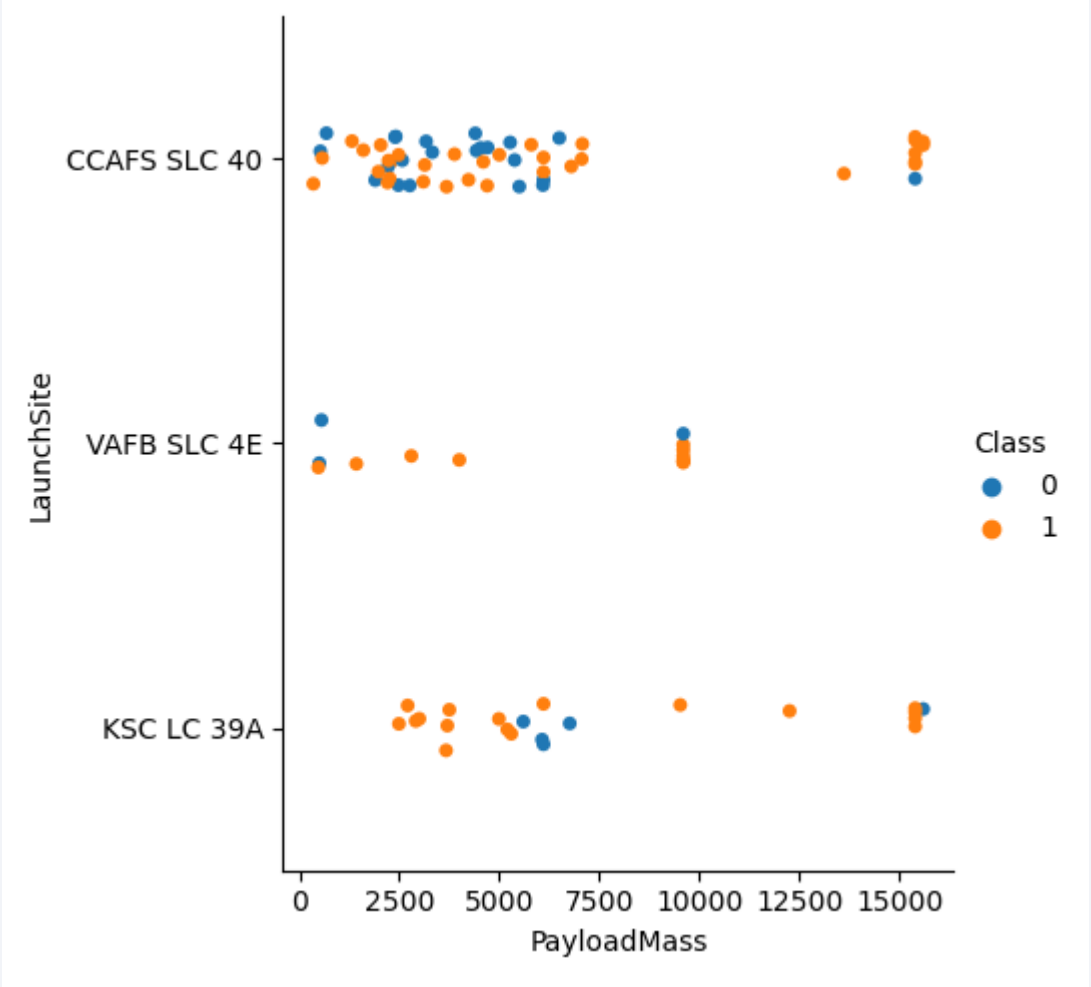
Flight Number vs. Launch Site

- For latter flights, the success rate seems to increase for all launch sites especially VAFB SLC 4E and CCAFS SLC 40.
- This indicates/proves the old adage that is ‘success comes with time’



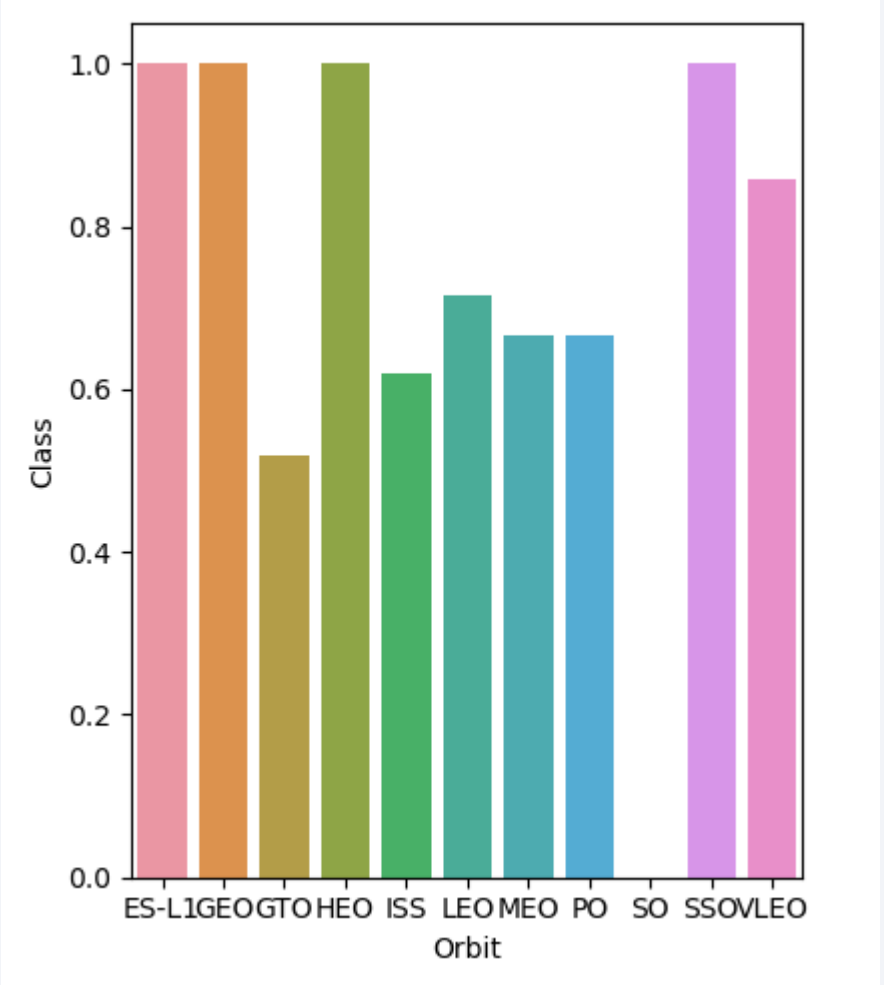
Payload vs. Launch Site

- For heavier payloads success rate seems to increase for CCAFS SLC 40 more specifically.
- For VAFB there weren't payloads greater than 10K lbs.



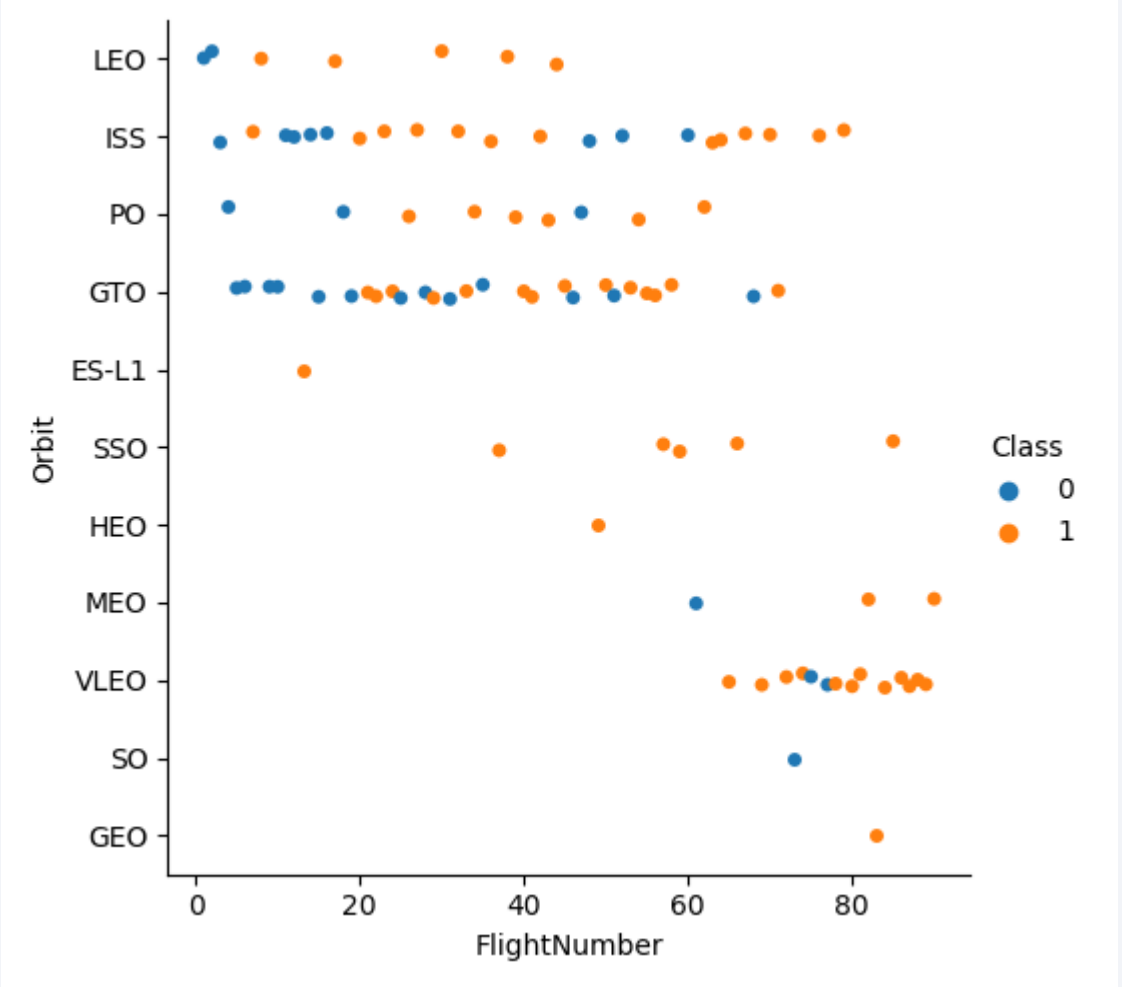
Success Rate vs. Orbit Type

- Orbits SSO, HEO, ES-L1 and GEO have had the highest success rates.
- Orbits such as GTO and ISS have had among the lowest success rates.



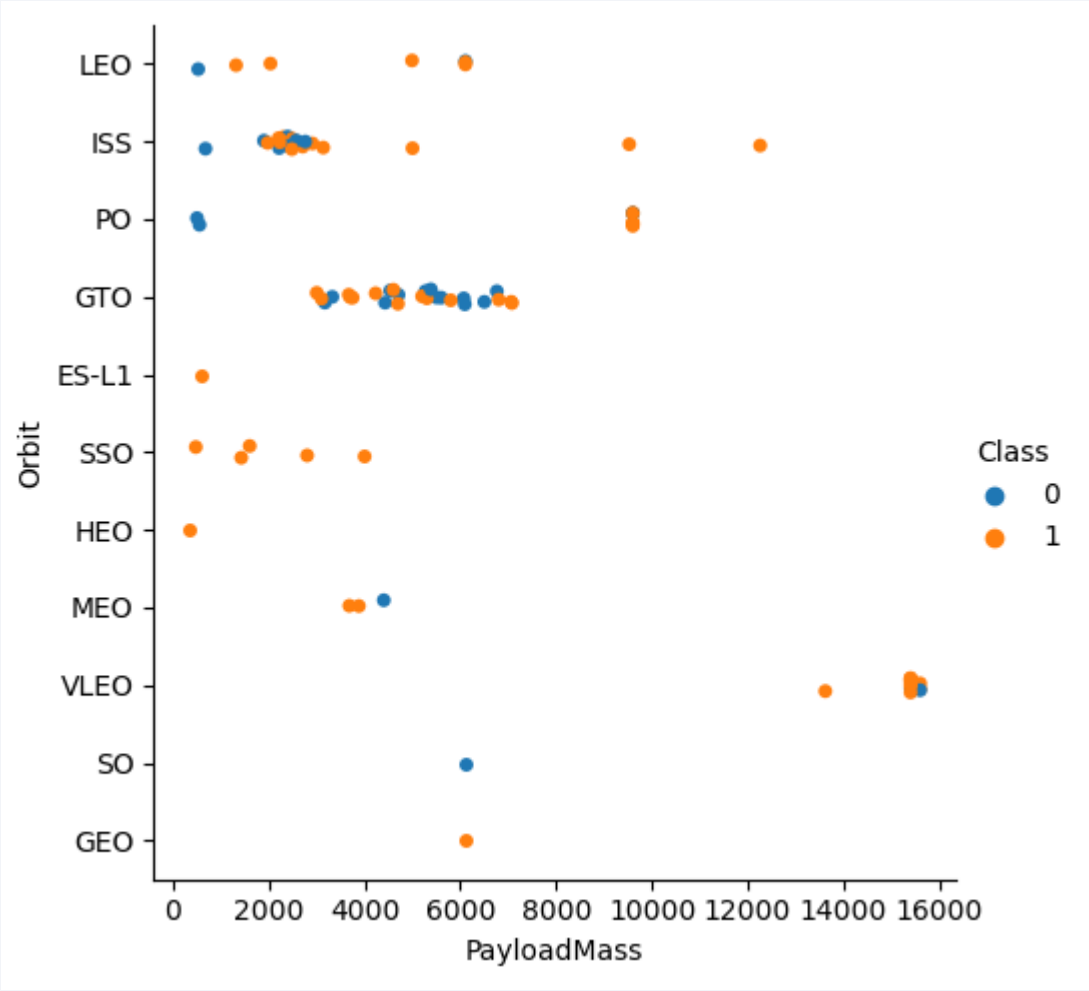
Flight Number vs. Orbit Type

- For the LEO orbit the Success appears to be related to the number of flights.
- On the contrast, there seems to be no relationship to the flight number for the GTO orbit.



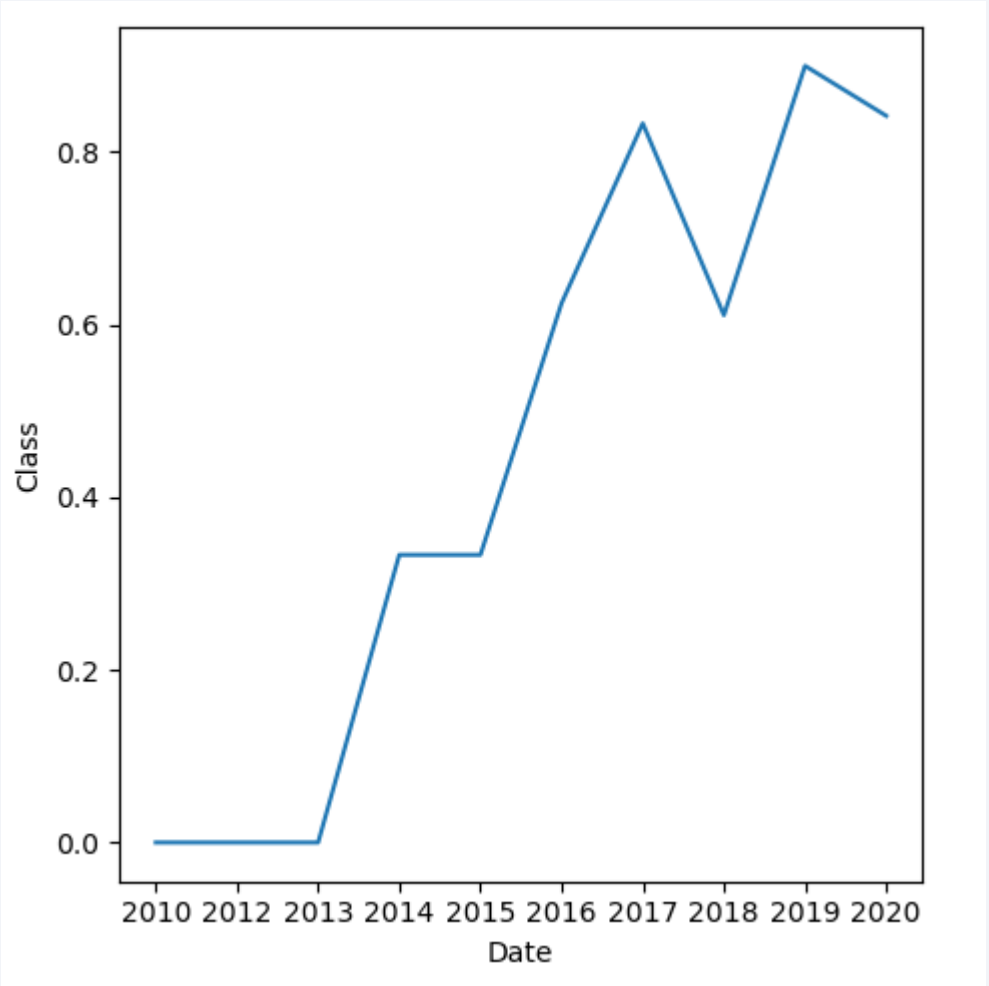
Payload vs. Orbit Type

- In the case of Po, LEO and ISS the heavier the load the higher the success rate.
- For GTO we cannot make such distinction. Too much entropy.



Launch Success Yearly Trend

- There was a spell of stagnancy at failure between 2010 and 2013
- The sucess rate since 2013 kept increasing till 2017.
- Decreased for one year and revamped at the end of 2018



All Launch Site Names

- To find the unique set of all launch sites we may deploy the following lines of code on the properly formatted dataframe.
  - launch_sites_df = spacex_df.groupby(['Launch Site'], as_index=False).first()
  - launch_sites_df = launch_sites_df[['Launch Site', 'Lat', 'Long']]

	Launch Site	Lat	Long
0	CCAFS LC-40	28.562302	-80.577356
1	CCAFS SLC-40	28.563197	-80.576820
2	KSC LC-39A	28.573255	-80.646895
3	VAFB SLC-4E	34.632834	-120.610745

Launch Site Names Begin with 'CCA'

- To find 5 records where launch sites begin with `CCA`, we may apply the following SQL query statement wrapped by the cursor object (API).

```
cur.execute ("SELECT Launch_Site FROM SPACEXTBL WHERE Launch_Site LIKE 'CCA%' LIMIT  5")
cur.fetchall()
```

```
[('CCAFS LC-40',),
 ('CCAFS LC-40',),
 ('CCAFS LC-40',),
 ('CCAFS LC-40',),
 ('CCAFS LC-40',)]
```

Total Payload Mass

- To calculate and display the total payload mass carried by boosters launched by NASA (CRS) for instance, we may write the following query statement.

```
cur.execute("SELECT SUM(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE SPACEXTBL.payload LIKE '%CRS%'")
print(cur.fetchall(), "Kg")
```

```
[(111268.0,)] Kg
```

Average Payload Mass by F9 v1.1

- To calculate and display the average payload mass carried by booster version F9 v1.1, we may use a LIKE statement as such.

```
cur.execute("SELECT AVG(PAYLOAD_MASS__KG_) FROM SPACEXTBL WHERE SPACEXTBL.Booster_Version LIKE '%F9 v1.1%'")
print("average payload mass carried by booster version F9 v1.1:" , cur.fetchall(), "Kg")
```

```
average payload mass carried by booster version F9 v1.1: [(2534.6666666666665,)] Kg
```

First Successful Ground Landing Date

- To find the date when the first succesful landing outcome in ground pad was achieved we may use the MIN function of SQL as such.

```
cur.execute("SELECT MIN(Date) FROM SPACEXTBL WHERE SPACEXTBL.Landing_Outcome LIKE '%Success (ground pad)%'")
cur.fetchall()
```

```
[('01/08/2018',)]
```

Successful Drone Ship Landing with Payload between 4000 and 6000

- To list the names of boosters which have successfully landed on drone ship and had payload mass greater than 4000 but less than 6000 …

```
cur.execute("SELECT Booster_Version FROM SPACEXTBL WHERE SPACEXTBL.Landing_Outcome LIKE '%Success (drone ship)%'
            AND 4000 <= SPACEXTBL.PAYLOAD_MASS__KG_  <= 6000 ")
cur.fetchall()
```

```
[('F9 FT B1021.1',),
, ('F9 FT B1022',),
, ('F9 FT B1023.1',),
, ('F9 FT B1026',),
, ('F9 FT B1029.1',),
, ('F9 FT  B1021.2',),
, ('F9 FT  B1029.2',),
, ('F9 FT B1036.1',),
, ('F9 FT B1038.1',),
, ('F9 B4 B1041.1',),
, ('F9 FT  B1031.2',),
, ('F9 B4 B1042.1',),
, ('F9 B4 B1045.1',),
, ('F9 B5  B1046.1',)]
```

Total Number of Successful and Failure Mission Outcomes

- To calculate the total number of successful and failure mission outcomes, we first fetch all mission outcomes and count them

```
cur.execute("SELECT DISTINCT(Mission_Outcome) FROM SPACEXTBL")
cur.fetchall()
```

```
[('Success',),
 ('Failure (in flight)',),
 ('Success (payload status unclear)',),
 ('Success ',),
 (None,)]
```

```
cur.execute("SELECT COUNT(Mission_Outcome) FROM SPACEXTBL")
cur.fetchall()
```

```
[(101,)]
```

Boosters Carried Maximum Payload

- To list the names of the boosters which have carried the maximum payload mass query as such.

```
cur.execute("SELECT Booster_Version, PAYLOAD_MASS__KG_ FROM SPACEXTBL
            WHERE PAYLOAD_MASS__KG_ = (SELECT MAX(PAYLOAD_MASS__KG_) FROM SPACEXTBL)")
cur.fetchall()
```

```
[('F9 B5 B1048.4', 15600.0),
, ('F9 B5 B1049.4', 15600.0),
, ('F9 B5 B1051.3', 15600.0),
, ('F9 B5 B1056.4', 15600.0),
, ('F9 B5 B1048.5', 15600.0),
, ('F9 B5 B1051.4', 15600.0),
, ('F9 B5 B1049.5', 15600.0),
, ('F9 B5 B1060.2 ', 15600.0),
, ('F9 B5 B1058.3 ', 15600.0),
, ('F9 B5 B1051.6', 15600.0),
, ('F9 B5 B1060.3', 15600.0),
, ('F9 B5 B1049.7 ', 15600.0)]
```

2015 Launch Records

- To list the failed landing_outcomes in drone ship, their booster versions, and launch site names for in year 2015, we query as following.

```
cur.execute("SELECT SUBSTR(Date,4,2) Landing_Outcome, Booster_Version, Launch_Site FROM SPACEXTBL AS SPX
            WHERE SUBSTR(SPX.Date,7,4) = '2015' AND SPX.Landing_Outcome LIKE '%Failure%' ")
cur.fetchall()
```

```
[('10', 'F9 v1.1 B1012', 'CCAFS LC-40'),
, ('04', 'F9 v1.1 B1015', 'CCAFS LC-40')]
```

Rank Landing Outcomes Between 2010-06-04 and 2017-03-20

- To rank the count of landing outcomes (such as Failure (drone ship) or Success (ground pad)) between the date 2010-06-04 and 2017-03-20, in descending order

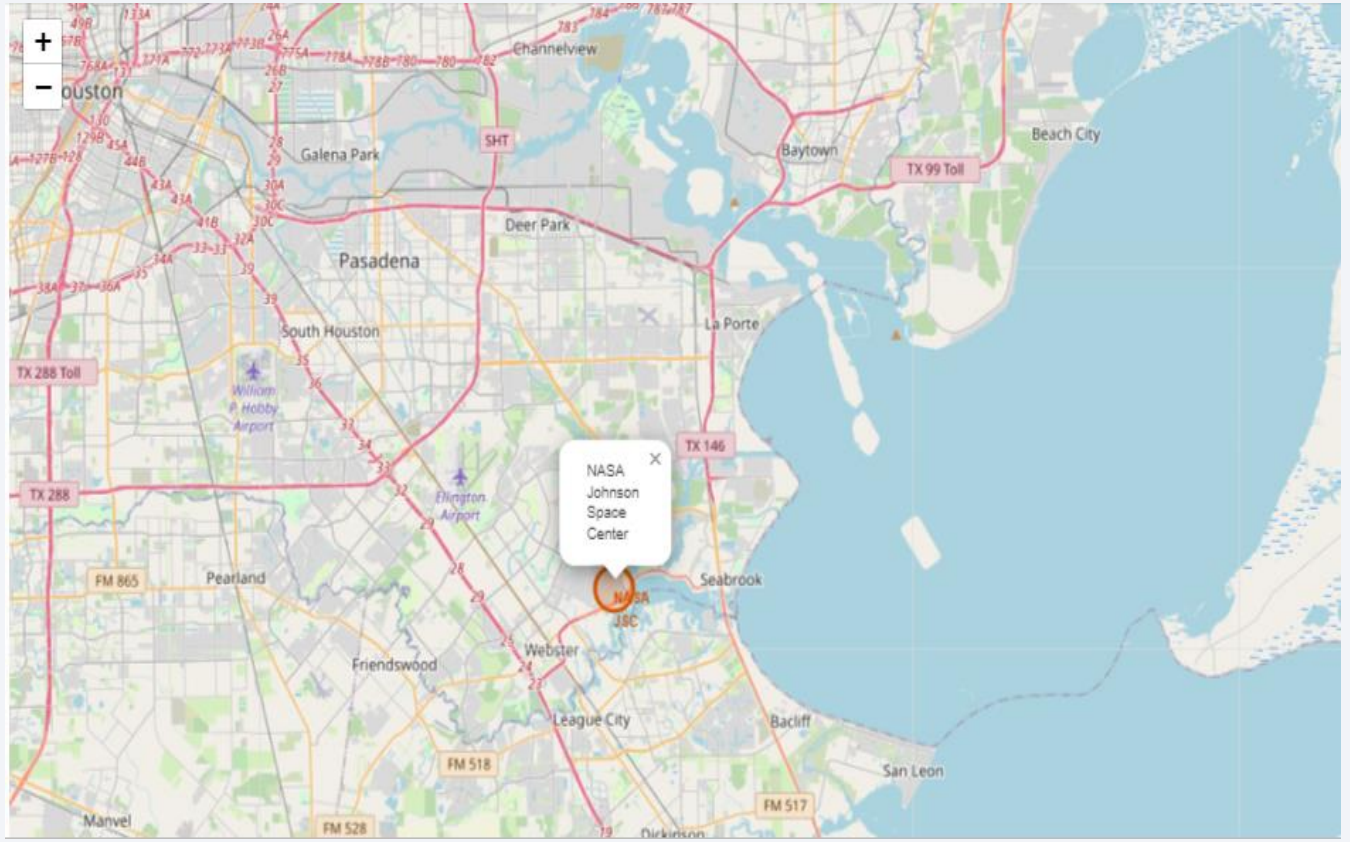
```
cur.execute("SELECT COUNT(Mission_Outcome) FROM SPACEXTBL WHERE 2010-06-04 <= Date <= 2017-03-20 DSC")
cur.fetchall()
```

Section 3

Launch Sites Proximities Analysis

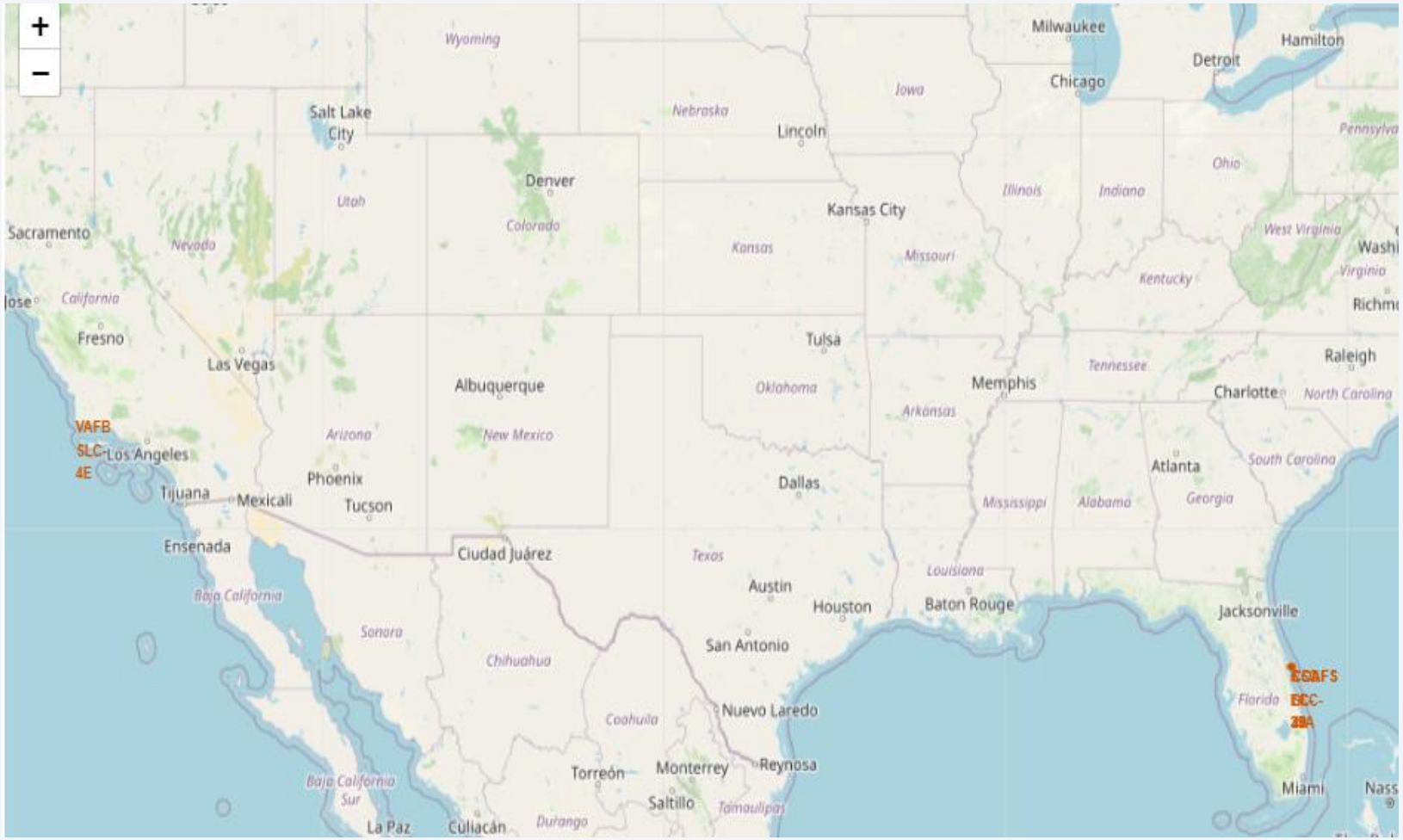
NASA Johnson Space Center

- Nasa JSC is used as focal point of the site map.
- Map opens to this location by default.



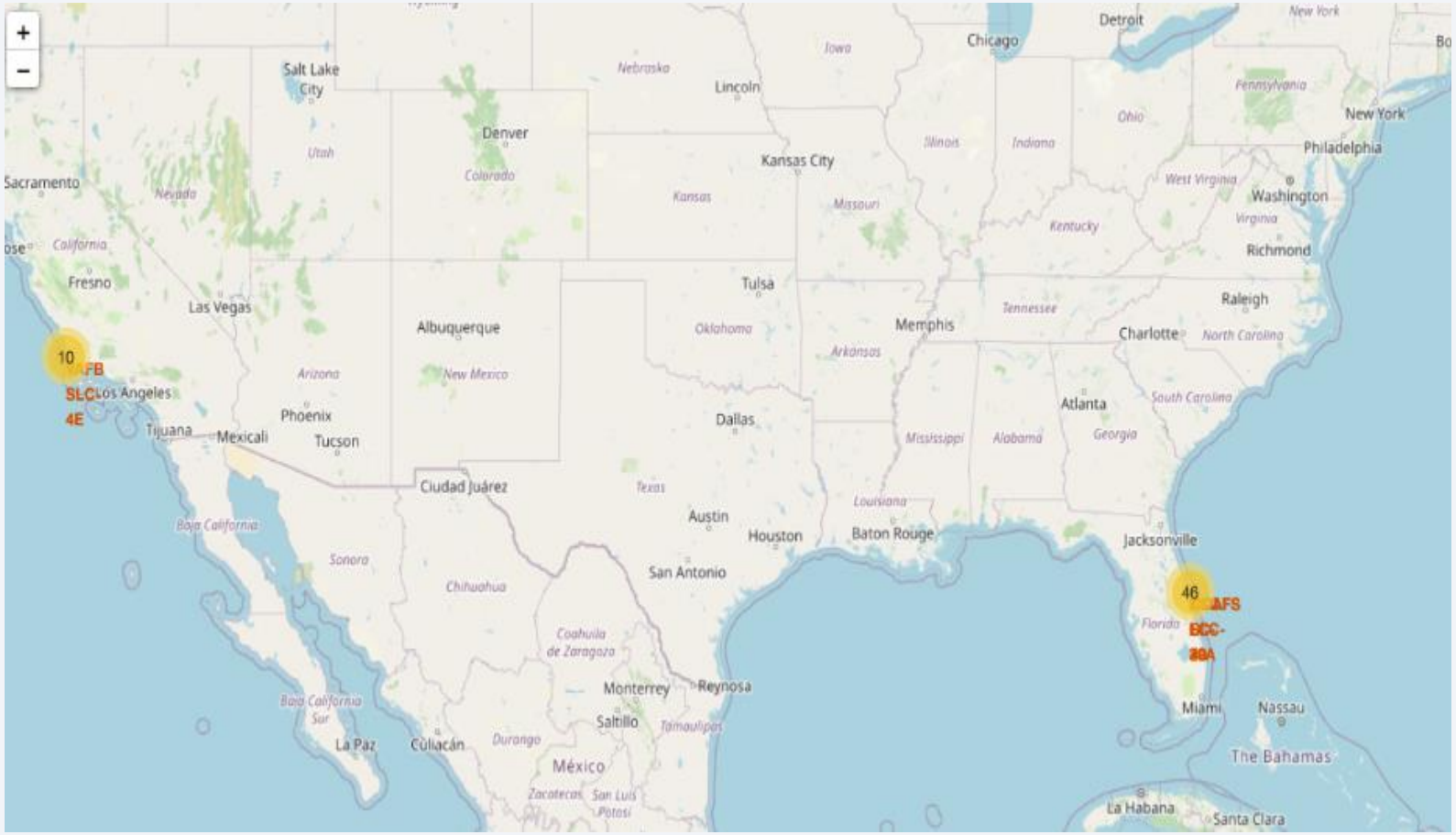
Launch Sites

- Created and added folium.Circle and folium.Marker for each launch site on the site map.
- We note that three of the four reside in Florida while one resides in california



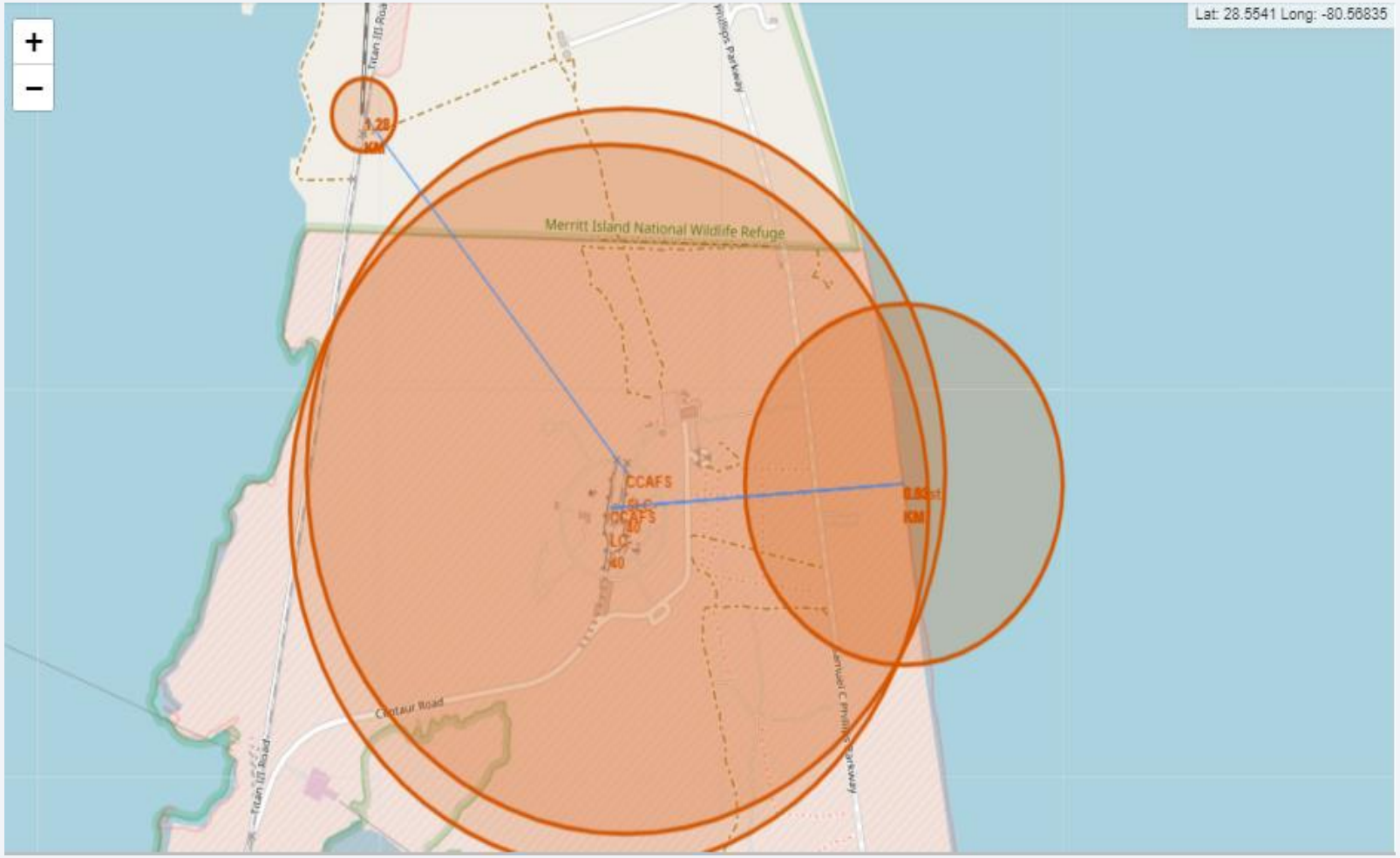
Marker Clusters

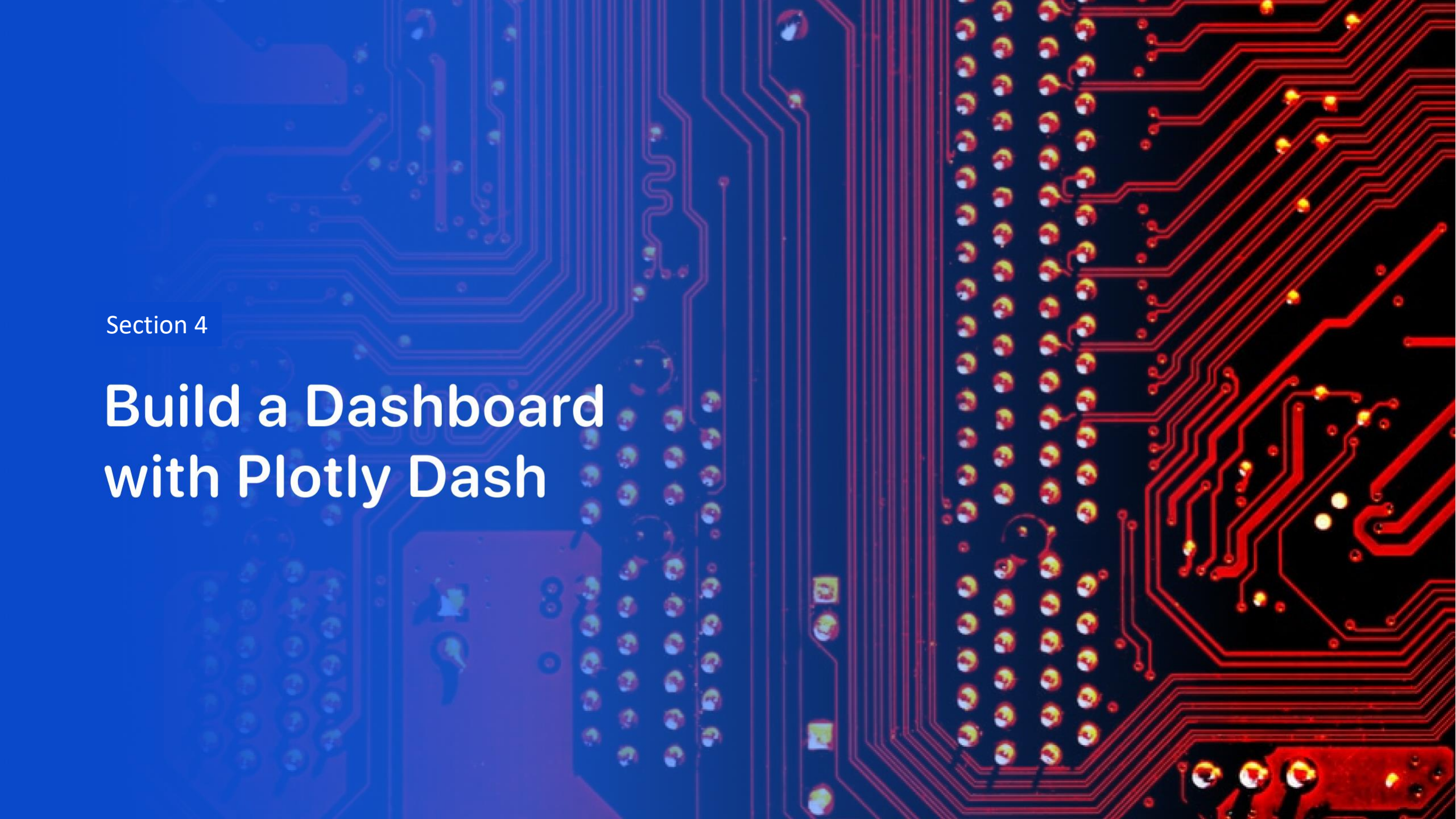
- Count the number of launches in each site and use marker cluster to store them
- As we zoom in the marker clusters will separate into their own categories by launch site.



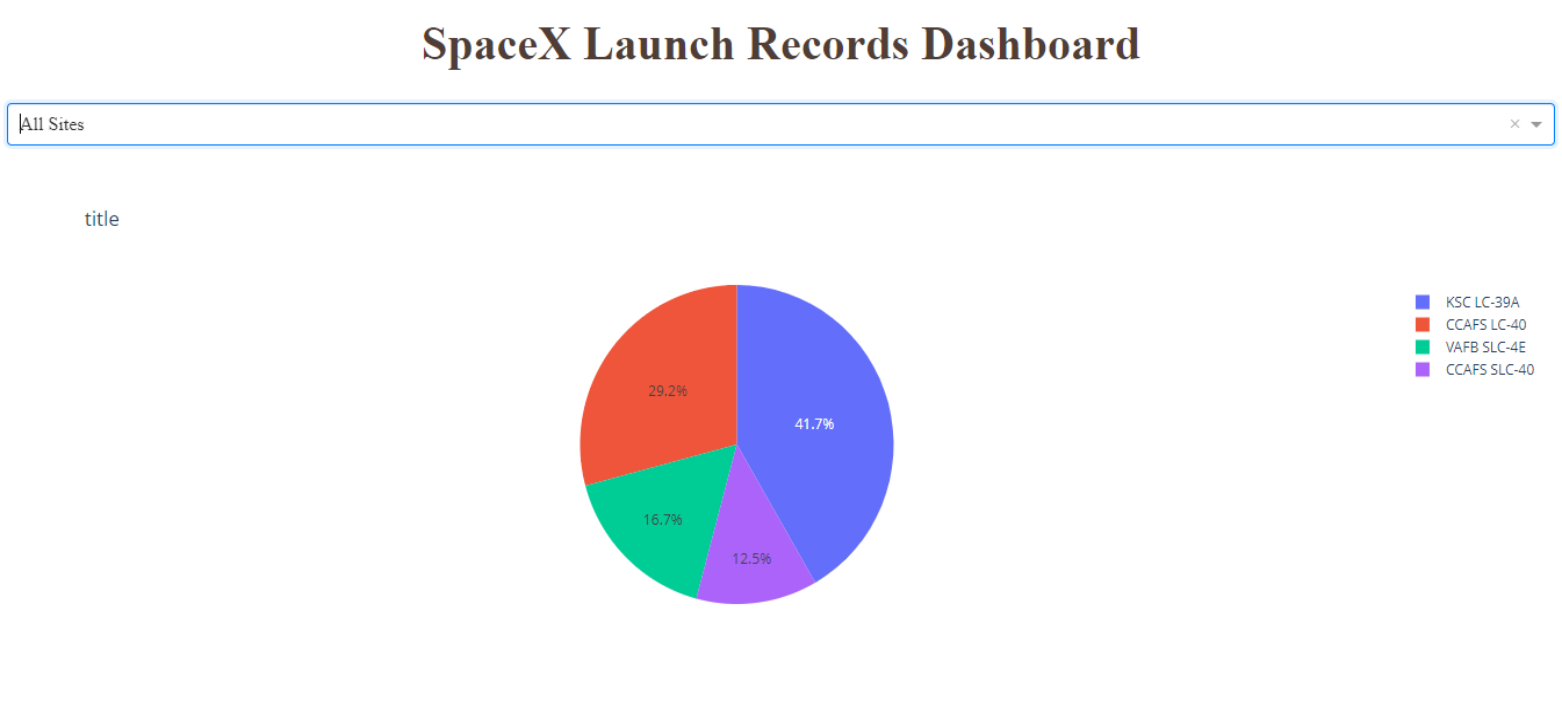
Launch site proximity to infrastructures and landmarks

- We choose a launch site and assess its access to nearby railroad and coast.
- We use a polyline and haversine functions to draw a line and find the distances.




Section 4
Build a Dashboard with Plotly Dash

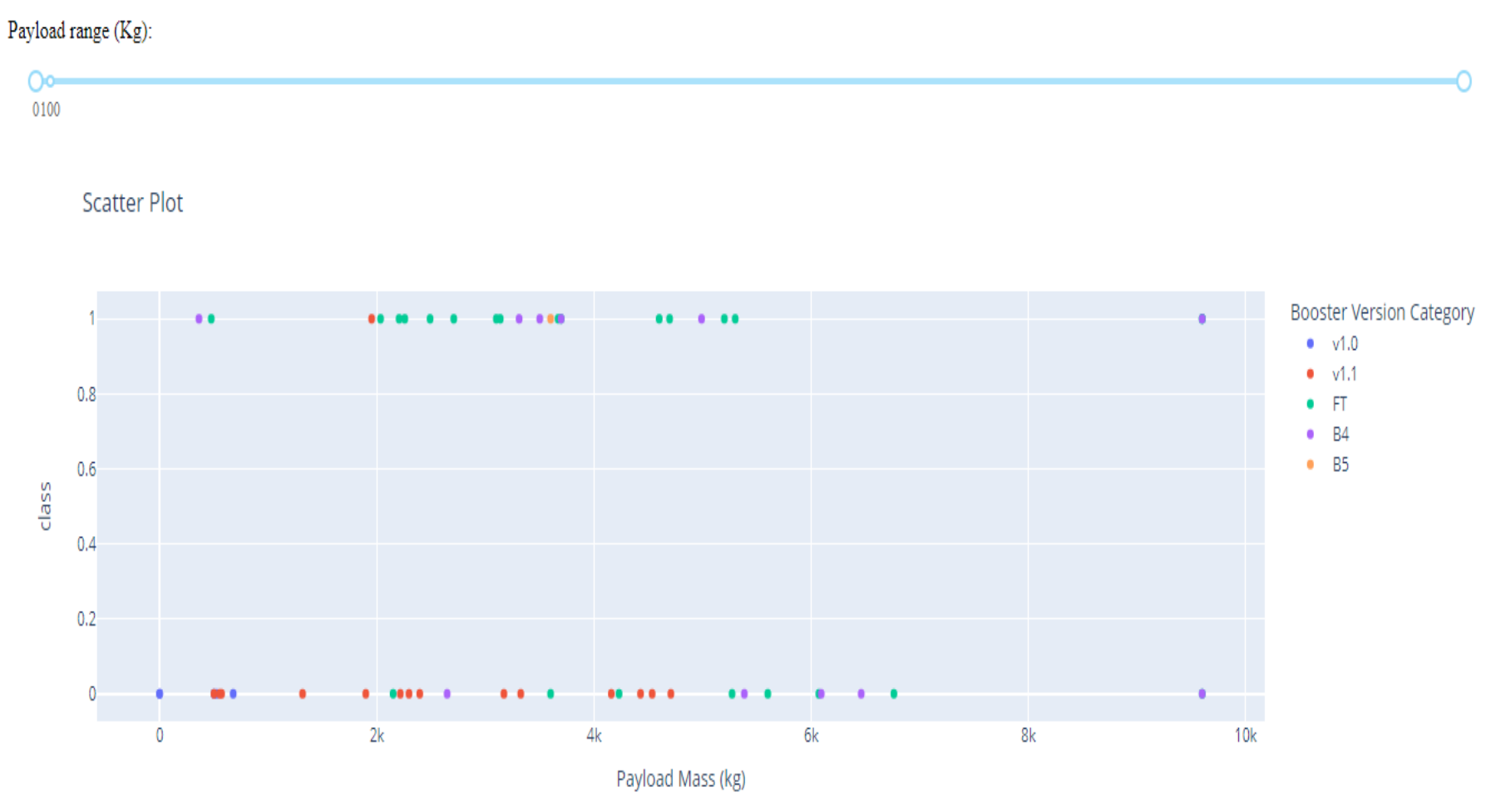
Launch sites classified by the number of Flights



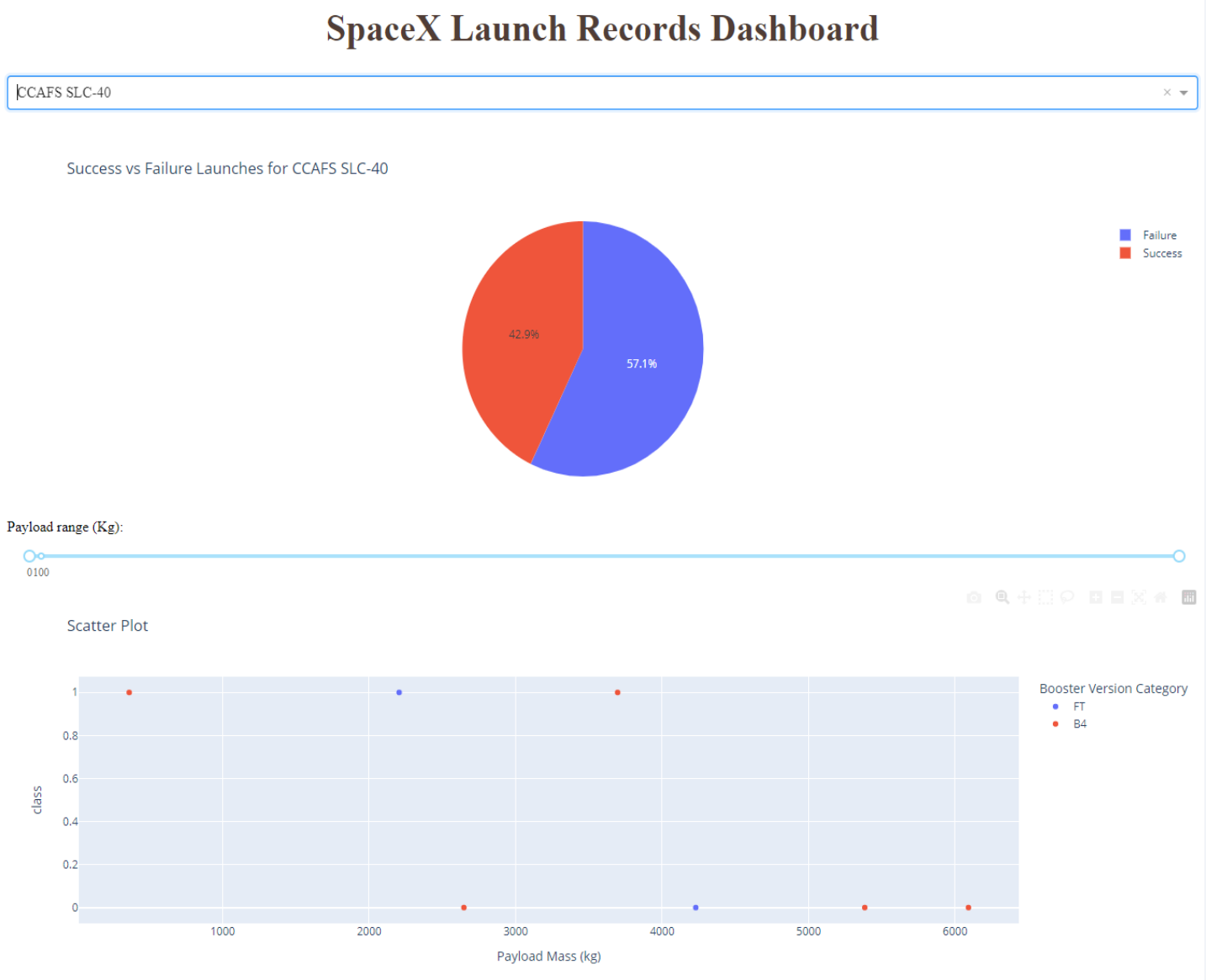
- All launch sites are represented in a pie chart by the number of flights they hosted as a percentage of the whole number of flights.
- Launch site names were used to color each piece on the pie chart.

Payload(Kg) vs Landing Outcome

- For all launch sites we graph an aggregate chart of the payload in Kg to the mission outcome.



Payload vs success for a selected site



- For any given site we plot the payload mass to the success rate.

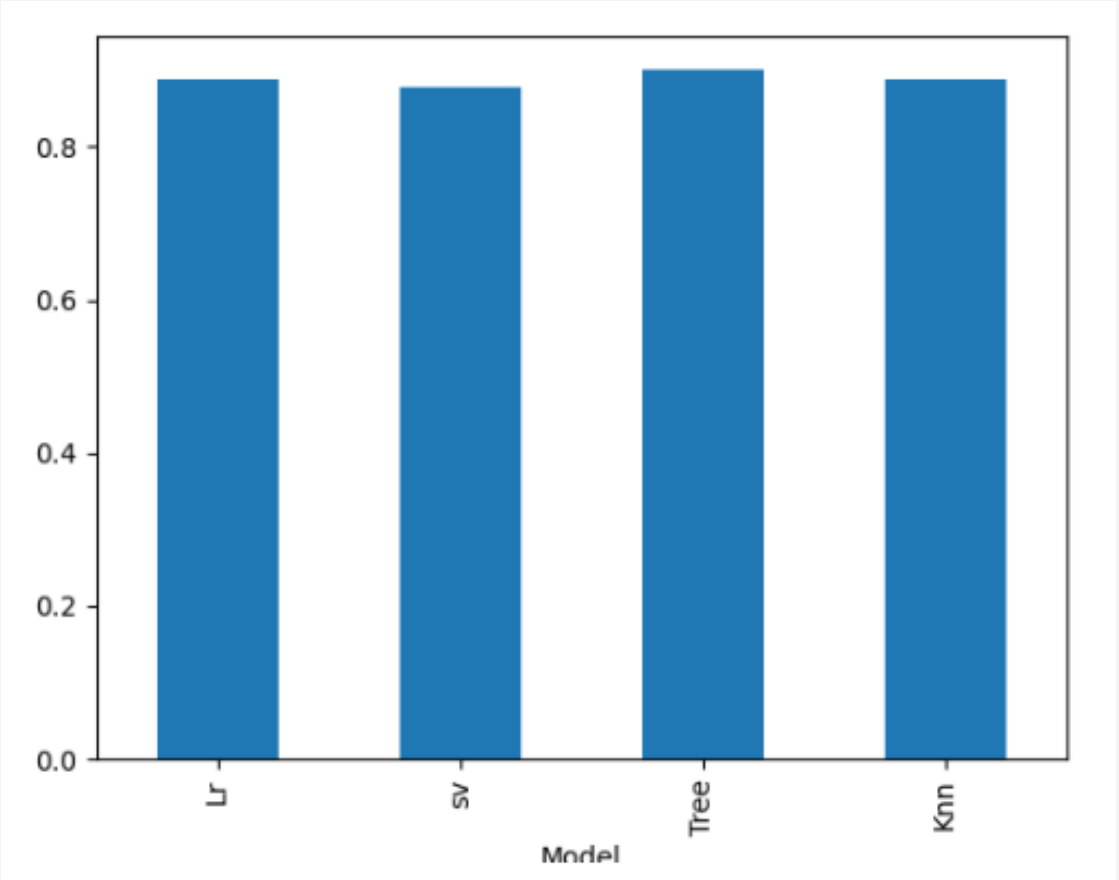
Section 5

Predictive Analysis (Classification)

Classification Accuracy of Different models

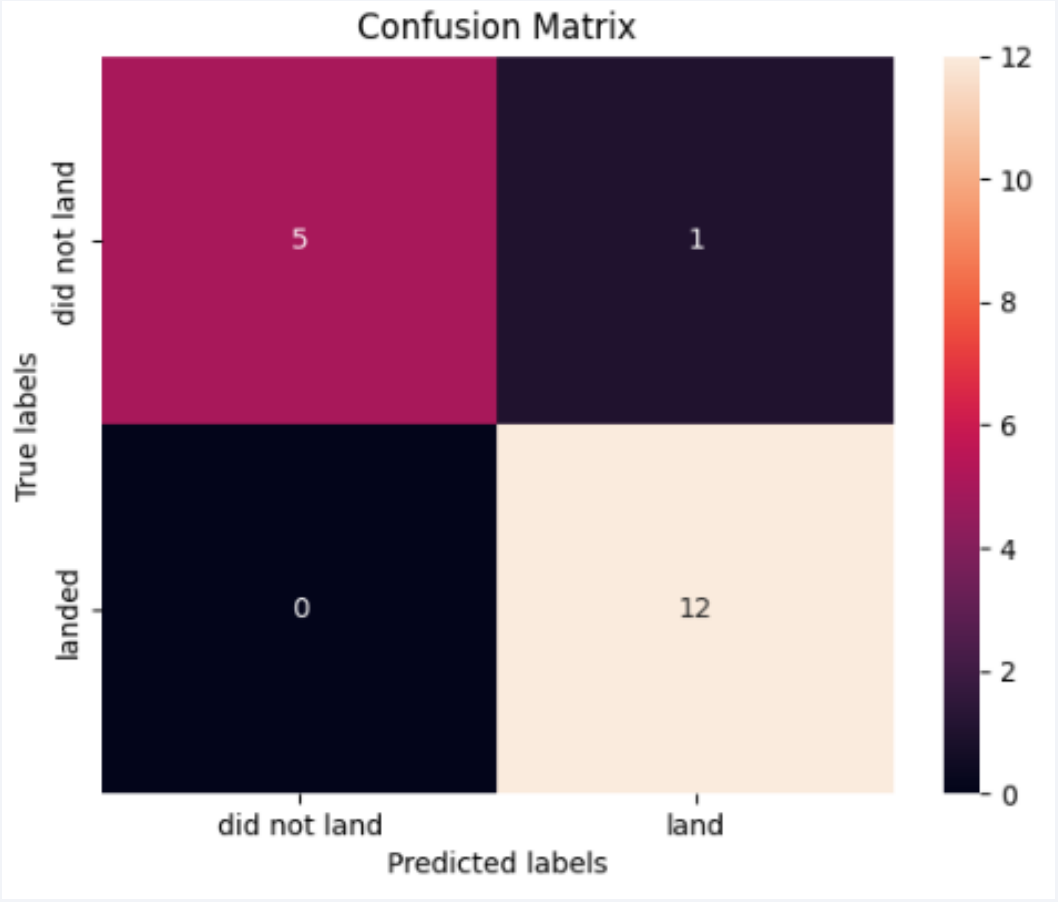
- Based on the tree score method applied to the GridSeachCV instance of each model we find that the tree classifier had the highest score.

- Logistic regression = 0.89,
- Support vector machines = 0.88
- Decision Tree Classifiers = 0.9
- K Nearest Neighbors= 0.89



Confusion Matrix of Decision Tree Classifier

- Best Estimator: DecisionTreeClassifier (criterion='entropy', max_depth=6, max_features='sqrt', min_samples_leaf=2).
- 12 true positves, 0 false negatives, 5 true negatives and 1 false positive predictions were made by the model.
- The model is near perfect



Conclusions

- The exploratory data analysis revealed meaningful relationships between the features in the data and helped determine what would be the label for training supervised models.
- The SQL and visual aid greaty enhanced the exploratory data analysis by helping us understand some details about key events and features.
- Took visual aid one step further by building interactive maps, plots and charts which brought words and numbers to life.
- Applied machine learning libraries to build, tune and evaluate classification models. Revealed Decision trees classifier as the best estimator based on a scoring method and confusion matrix.

Appendix

- CSV file
- https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_2.csv
- CSV file with one hot encoded data
- https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DS0321EN-SkillsNetwork/datasets/dataset_part_3.csv

Thank you!